Freie Universität Berlin
Fachbereich Mathematik

Diplomarbeit

Classification of smooth lattice polytopes
with few lattice points

Supervisor: Dr. Christian Haase

Abstract: After giving a short introduction on smooth lattice polytopes, I will present a proof for the finiteness of smooth lattice polytopes with few lattice points. The argument is then turned into an algorithm for the classification of smooth lattice polytopes in fixed dimension with an upper bound on the number of lattice points. Additionally I have implemented this algorithm for dimension two and three and used it, together with a classification of smooth minimal fans by Tadao Oda, to create lists of all smooth 2-polytopes and 3-polytopes with at most 12 lattice points.

by: Benjamin Lorenz
lorenz.benjamin@fu-berlin.de

Contents

1. Introduction

1.1. Polytopes and Lattices

Our main goal is the classification of smooth lattice polytopes with few lattice points, so we will start by introducing polytopes and lattice polytopes.

Definition. A set $P \subset \mathbb{R}^d$ is called a **polytope** if it is the convex hull of a finite set of points $V = \{v_1, \ldots, v_n\} \subset \mathbb{R}^d$. If V is minimal up to inclusion, then the points $v_1, \ldots, v_n$ are the vertices of P.

$$P = \text{conv}(V) = \left\{ \sum_{v \in V} \lambda_v v \in \mathbb{R}^d \mid \sum_{v \in V} \lambda_v = 1, \lambda_v \geq 0 \ \forall v \in V \right\}$$

Remark. In the following we will always assume V to be the vertex set of the polytope.

For the definition of smooth lattice polytopes, and to make precise what we mean by polytopes with few lattice points, we first need to define lattices in $\mathbb{R}^d$.

Definition. For some finite set $B = \{b_1, \ldots, b_n\} \subset \mathbb{R}^d$, we define the **lattice** $\Lambda(B)$:

$$\Lambda(B) = \left\{ \sum_{b \in B} \lambda_b b \in \mathbb{R}^d \mid \lambda_b \in \mathbb{Z} \ \forall b \in B \right\} \subset \mathbb{R}^d$$

If the vectors in B are linearly independent then B is called a **basis** of $\Lambda(B)$.

Remark. We will now restrict to the lattice $\Lambda = \mathbb{Z}^d \subset \mathbb{R}^d$, so any set $B = \{b_1, \ldots, b_d\} \subset \mathbb{Z}^d$ is a basis for $\mathbb{Z}^d$ if $\det(B) = \pm 1$.

Now, for a lattice $\Lambda = \Lambda(B) \subset \mathbb{R}^d$, the **dual lattice** is

$$\Lambda^* = \{\gamma \in (\mathbb{R}^d)^* \mid \gamma(x) \in \mathbb{Z} \ \forall x \in \Lambda\}$$

and one can easily see that for $\Lambda(B) = \mathbb{Z}^d$ we have $\Lambda^* = \Lambda((B^{-1})^T) \cong \mathbb{Z}^d$. So for some lattice basis B define the **dual lattice basis** $B^* = (B^{-1})^T$.

We can now define smooth lattice polytopes:

Definition. Let P be a polytope. Then P is called a **lattice** polytope if all vertices have integer coordinates, i.e. $V \subseteq \mathbb{Z}^d$.

When we talk about the lattice points $L(P)$ of a polytope $P \subset \mathbb{R}^d$ we mean the elements of $\mathbb{Z}^d$ that lie in the polytope, namely $L(P) = P \cap \mathbb{Z}^d$. So we want to limit the number $|L(P)|$ for our polytopes.

For a lattice polytope P to be smooth it has to be simple. So for $v \in V$ denote by $\text{nb}(v)$ the set of adjacent vertices, i.e. for every $w \in \text{nb}(v)$ there is an edge containing v and w. Then a polytope $P \subset \mathbb{R}^d$ is **simple** if $|\text{nb}(v)| = d$ for all vertices $v \in V$.

Definition. A simple lattice polytope P is called **smooth** if, at each vertex, the lattice minimal edge-directions form a lattice basis, or more formally

$$\det \left(\left\{ \frac{w - v}{\gcd(w - v)} \mid w \in \mathrm{nb}(v) \right\} \right) = \pm 1 \quad \forall v \in V$$

1.2. Cones and Fans

In our main algorithm we want to approach the polytopes from the dual and therefore we now introduce cones and fans and define some important properties.

Definition. For $S \subset \mathbb{R}^d$ finite, the **polyhedral cone** $C = \mathrm{cone}(S)$ is

$$\mathrm{cone}(S) = \left\{ \sum_{r \in S} a_r r \in \mathbb{R}^d \mid a_r \in \mathbb{R}_{\geq 0} \ \forall r \in S \right\}$$

A cone $C \subset \mathbb{R}^d$ is called **pointed** if there is $\alpha \in (\mathbb{R}^d)^*$ such that

$$C \cap \{x \in \mathbb{R}^d \mid \alpha(x) \leq 0\} = \{0\}$$

If $C = \mathrm{cone}(S)$ is pointed and S is minimal up to inclusion, then the vectors $r \in S$ are called the **rays** of C.

Remark. Some restrictions on the cones we will consider later on.

- All our cones are pointed.

- All our cones are rational, which means that there is $S \subset \mathbb{Q}^d$ finite, such that $C = \mathrm{cone}(S)$.

- And we assume all rays $r \in S$ to be primitive vectors, i.e. $\gcd(r) = 1$ for all $r \in S$.

The **dimension** of a cone $C = \mathrm{cone}(S)$ is the dimension of its linear hull $\dim(C) = \dim(\mathrm{span}(S))$. A k-dimensional cone C is called **simplicial** if there is $S = \{r_1, \ldots, r_k\}$ such that $C = \mathrm{cone}(S)$. One may already see the corresondence to simple polytopes. We will continue in this direction by defining smooth cones. Later we will see that both definitions of smoothness coincide if we choose the right construction to define a fan from a polytope.

Definition. Let $C \subset \mathbb{R}^d$ be a k-dimensional simplicial cone, then C is **smooth** if there are $z_1, \ldots, z_d \in \mathbb{Z}^d$ such that

- $C = \mathrm{cone}(\{z_1, \ldots, z_k\})$

- $\det(z_1, \ldots, z_d) = \pm 1$

Definition. A **fan** F is a finite set of cones with the following properties.

(1) $C_1, C_2 \in F \Rightarrow C_1 \cap C_2 \in F$

(2) $C \in F$, $C' \neq \emptyset$, $C' \leq C$ (i.e. C' is a non-empty face of C) $\Rightarrow C' \in F$

For a fan F, denote by $F^{(k)}$ the set of cones of dimension k. A fan F in $\mathbb{R}^d$ is called **complete** if it covers the whole $\mathbb{R}^d$, which means $\bigcup_{C \in F} C = \mathbb{R}^d$.
The properties of cones defined above can of course be extended to fans, a fan is **simplicial** or **smooth** if all its cones are.

1.3. Duality

In the third section of the introduction we will now state a duality theorem of polyhedral theory and establish a connection between polytopes and fans. This will turn out useful later in the classification.

Definition. For a matrix $A \in \mathbb{R}^{m \times d}$ and a vector $b \in \mathbb{R}^m$ the **polyhedron** $P(A, b) \subseteq \mathbb{R}^d$ is the intersection of finitely many halfspaces

$$P(A, b) = \bigcap_{a_i \text{ row of } A} \{x \in \mathbb{R}^d | \langle a_i, x \rangle \leq b_i\} = \{x \in \mathbb{R}^d | Ax \leq b\}$$

For the duality theorem we need the **Minkowski sum** $P + Q$ of two subsets $P, Q \subset \mathbb{R}^d$:

$$P + Q = \{x + y \in \mathbb{R}^d \mid x \in P, y \in Q\}$$

Theorem. *A set $P \subseteq \mathbb{R}^d$ is a polyhedron if and only if it is the Minkowski sum of a polytope and a cone.*

Proof. The proof of this duality can be found in most textbooks containing polyhedral theory, e.g. *Lectures on Polytopes* [8].

Remark. In particular we have the following results for polytopes:

- A polytope is a bounded polyhedron.

- For $P = P(A, b)$ bounded, there is $V \subset \mathbb{R}^d$ finite, such that $P = \mathrm{conv}(V)$.

- And for $P = \mathrm{conv}(V)$ there are $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$ such that $P = P(A, b)$.

This is very useful as it allows us to switch between both representations as we need it. Now let us define the normal fan of a polytope, the main construction we will use for the classification of the polytopes.

Definition. Let $P = P(A, b) \subset \mathbb{R}^d$ be a polytope, the **normal cone** of a face $F \subset P$ is

$$\mathrm{NC}_P(F) = \mathrm{cone}(\{a_i \text{ row of } A \mid \langle a_i, x \rangle = b_i \ \forall x \in F\})$$

The **normal fan** of P is defined by the normal cones of all faces of the polytope.

$$\mathrm{NF}_P = \{\mathrm{NC}_P(F) \mid F \subset P \text{ face}\}$$

To see that this does indeed produce a fan, observe that for every face F of the polytope the faces containing F are defined by subsets of the rays defining F. Thus, the faces of P containing F define the faces of $\mathrm{NC}_P(F)$. Additionally this construction always produces a complete fan.

This yields a construction to build a fan from a given polytope. We can also use this in the other direction. For a given matrix A containing the rays of the fan we can define a polytope $P(A, b)$ by choosing a right-hand side vector b. We just need to make sure that the normal fan of the polytope is again the fan we started with, since there can be different polytopes with the same set of facet normals.
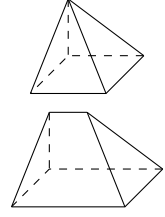
However, we only want to classify smooth polytopes, so the obvious question is whether there is a property of the normal fan that coincides with the smoothness of the polytope. And the obvious answer is also correct:

Proposition. *Let P be lattice d-polytope and NF_P its normal fan. Then P is smooth if and only if NF_P is smooth.*

Proof. First we will show that P is simple if and only if NF_P is simplicial.

- Let NF_P be a simplicial fan, then the normal cone of a vertex v of P is defined by exactly d rays. Each $(d-1)$ subset of the facets defined by these rays can define an edge and since there are d such subsets, there can be at most d edges containing v. Because P is a d-dimensional polytope, there must be at least d edges containing v and thus P is simple.

- Let P be a simple polytope and v be a vertex of P. Now, every facet containing v is itself a $(d - 1)$-dimensional polytope and thus there must be at least $(d - 1)$ edges containing v in each facet. Since there are only d $(d - 1)$-subsets of the d edges at v there can be at most d different facets and thus NF_P is simplicial.

We choose again one vertex v in P. To prove the smoothness we use the matrizes E, A defined by the edge-directions $e_1, \dots, e_d$ at v and the normal cone of the vertex $N_P(v) = \mathrm{cone}(\{a_1, \dots, a_d\})$.

For given smooth edge-directions E choose $A = -(E^{-1})^T$, obviously A defines the normal cone. By Cramer's rule and since $\det(E) = \pm 1$ we know that A is integral and $\det(A) = \mp 1$, thus the cone is smooth. And for given smooth normal fan with rays in A choose $E = -(A^{-1})^T$. By the same argument $E \in \mathbb{Z}^{d \times d}$, $\det(E) = \mp 1$ and thus E contains the lattice minimal edge-directions. $\qquad\square$

2. Finiteness

2.1. Finiteness Theorem

Now we can state the main theorem [5].

Theorem. *For integers d and N, there are only finitely many smooth d-polytopes with at most N lattice points.*

Since the following proof inspired the technique for the classification algorithm following subsequently, we will repeat its details.

Proof. We will proof this in three steps. First we want to limit the number of combinatorial types of smooth fans, then the number of smooth fans with each of those types and finally the number of smooth polytopes having one of those as normal fan, all using the limit on the number of lattice points.
In the following we will again assume all rays to be primitive.

- Assume we have a combinatorial type of a complete simplicial fan F with n rays and at most N full-dimensional cones. A full-dimensional cone of this fan can be defined by d-subsets $C \subset \{1, \ldots, n\}$, $|C| = d$. The fan can then be defined by the set of all such full-dimensional cones $F^{(d)} \subset \binom{\{1,\ldots,n\}}{d}$. *Note that we use a different notation here since we consider only the combinatorial type of the fan and do not care about the rays.* We know that the fan has at most N full-dimensional cones, so $|F^{(d)}| \leq N$. Additionally every ray needs to appear in at least d full-dimensional cones:

$$d_i = |\{C \in F^{(d)} \mid i \in C\}| \geq d \qquad \forall i \in \{1, \ldots, n\}$$

By counting all appearances of all rays in $F^{(d)}$ in two different ways we have:

$$d \cdot n \leq \sum_{i=1}^{n} d_i \leq d \cdot N$$

And thus $n \leq N$, which means that we have at most N rays in the simplicial fan F. Hence our fan is fully defined by choosing

$$F^{(d)} \subset \binom{\{1, \ldots, n\}}{d} \subset \binom{\{1, \ldots, N\}}{d}$$

Since N and d are fixed there are only finitely many such subsets, and thus only finitely many possible combinatorial types of simplicial fans.

- Given the combinatorial type we now want to define the fan F up to isomorphism: Let $C = \text{cone}(n_1, n_2, \ldots, n_{d-1}) \in F$ be a codimension 1 cone which is contained in two full-dimensional cones $C_1 = \text{cone}(C, r_1)$ and $C_2 = \text{cone}(C, r_2)$. Since all the cones are

smooth r_1 and r_2 have lattice distance one from C, but lie on different sides of C. So $r_1 + r_2$ lies in the span of C and thus

$$r_1 + r_2 = \sum_{i=1}^{d-1} a_{C,i} n_i$$

The coefficients $a_{C,i}$ are the edge-paremeters of a fan F. We know $a_{C,i} \in \mathbb{Z}$ because C is smooth and thus $\{n_1, \ldots, n_{d-1}\}$ is a Hilbert basis of C.

We can also use the edge-parameters $a_{C,i}$ for $1 \le i \le d-1$, $C \in F^{(d-1)}$ together with the combinatorial type to reconstruct fan F using the following technique: Every smooth fan can, while preserving the lattice, be transformed to contain the cone defined by all d unit vectors. This means we can assume the fan to contain this cone. Then there are codimension 1 cones for every $(d-1)$-subset of the unit vectors, so, given the edge-parameters for each one of those cones, one can calculate the ray on the other side of this codimension 1 cone with the formula above. Iterate until all rays are calculated. Since the edge-parameters are in $\mathbb{Z}$ there may be infinitely many such smooth fans, but in the next step we will show a way to bound those parameters.

• In the last step we have to bound the number of smooth polytopes P with at most N lattice points having a given normal fan F and the number of smooth fans the above construction yields.

We therefore consider the edge e of the polytope dual to a codimension 1 cone $C \in F$. Let $l(e)$ be the lattice length of the edge e and x_1, x_2 be the vertices defining this edge. Denote by $u_1, \ldots, u_d$ the lattice-minimal edge-directions at x_1 which are dual to the inner normals $-n_1, \ldots, -n_{d-1}, -r_1$ around x_1. So we have $x_2 = x_1 + l(e)u_d$. Using the edge-directions at x_1 and the edge-parameters we can calculate the edge-directions at x_2.

Lemma. For x_1, x_2 as above, the edge-directions at x_2 are

$$u_i' = u_i + a_{C,i} u_d \quad \text{for } i \in \{1, \ldots, d-1\}$$
$$u_d' = -u_d$$

Proof. The inner normals around x_2 are $-n_1, \ldots, -n_{d-1}, -r_2$ and

$$-r_2 = r_1 - \sum_{i=1}^{d-1} a_{C,i} n_i$$

so we have

$$\langle -n_i, u_j' \rangle = \langle -n_i, u_j + a_{C,j} u_d \rangle = \langle -n_i, u_j \rangle - a_{C,j} \langle n_i, u_d \rangle \qquad \text{for } i, j < d$$
$$= \langle -n_i, u_j \rangle = \delta_{i,j} \qquad \text{for } i, j < d$$
$$\langle -n_i, u_d' \rangle = \langle -n_i, -u_d \rangle = 0 \qquad \text{for } i < d$$

8

$$\langle -r_2, u_j' \rangle = \langle r_1, u_j + a_{C,j} u_d \rangle - \sum_{i=1}^{d-1} a_{C,i} \langle n_i, u_j + a_{C,j} u_d \rangle \qquad \text{for } j < d$$

$$= a_{C,j} \langle r_1, u_d \rangle - a_{C,j} \langle n_j, u_j \rangle = 0 \qquad \text{for } j < d$$

$$\langle -r_2, u_d' \rangle = \langle r_1, -u_d \rangle - \sum_{i=1}^{d-1} a_{C,i} \langle n_i, -u_d \rangle = \langle -r_1, u_d \rangle = 1$$

which proves that the matrix containing these edge-directions is inverse to the given inner normals, finishing the proof of the lemma. $\qquad\square$

Since in our smooth lattice polytope every edge must have length at least 1 the lattice points defined by the vertex x_2 plus one of the edge-directions must of course lie in the polytope and thus

$$\langle r_1, x_2 + a_{C,i} u_d \rangle \geq \langle r_1, x_1 \rangle$$
$$\langle r_1, x_1 + l(e) u_d + a_{C,i} u_d \rangle \geq \langle r_1, x_1 \rangle$$
$$(l(e) + a_{C,i}) \langle r_1, u_d \rangle \geq 0$$
$$a_{C,i} \geq -l(e) \geq -N$$

This gives a lower bound for the edge-parameters. For an upper bound let us first introduce the thickened edge.
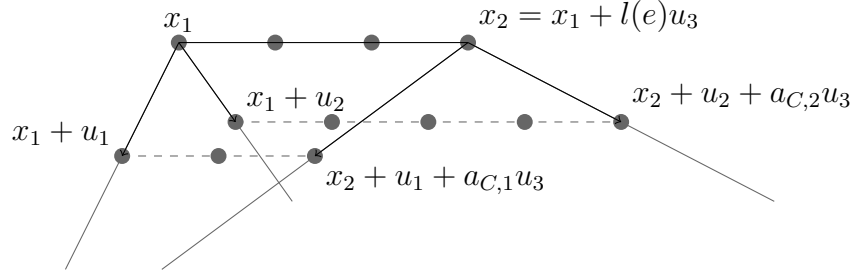
Figure 1: A thickened edge

Definition. Using the notation from above we define the thickened edge in a smooth polytope as the convex hull of all those points we just discussed.

$$c(e) = \text{conv}(\{x_1, x_1 + u_1, \ldots, x_1 + u_{d-1},$$
$$x_2, x_2 + u_1 + a_{C,1} u_d, \ldots, x_2 + u_{d-1} + a_{C,d-1} u_d\})$$

We obviously have $c(e) \subseteq P$ and as shown in Figure 1 $c(e)$ contains $d-1$ lines from $x_1 + u_i$ to $x_2 + u_i + a_{C,i} u_d$ each containing $l(e) + 1 + a_{C,i}$ lattice points and the line from x_1 to x_2 containing $l(e) + 1$ lattice points, so

$$|c(e) \cap \mathbb{Z}^d| = d(l(e) + 1) + \sum_{i=1}^{d-1} a_{C,i}$$

9

Since the edge length must be positive and we want the number of lattice points of P and thus of $c(e)$ to be less than N, this gives us an upper bound for the sum of all edge-parameters. Together with the lower bound this leaves us with only finitely many choices for the edge-parameters. This means for an upper bound N on the number of lattice points in the polytope there are only finitely many possible smooth normal fans. Now for a given normal fan we can define the polytope up to isomorphism by fixing all edge lengths. Just assume one vertex at the origin and the edge-directions to be the unit vectors, then the edge lengths define the next vertices and the normal fan defines the next edge-directions. Iterate until all vertices are fixed. And we want of course the edge length $l(e)$ to be positive and less than N, so for every edge there are only finitely many possible edge lengths. And thus only finitely many polytopes for this fan.

So altogether we have finitely many combinatorial types of fans, finitely many choices for the parameters and finitely many choices for the edge lengths and thus only finitely many smooth lattice d-polytopes with at most N lattice points. This completes the proof of the finiteness theorem. □

2.2. General Algorithm

The proof gives rise to the following approach for the classification of such polytopes.

2.2.1. Generating Normal Fans

The first step is to find all combinatorial types of complete simplicial fans in dimension d with at most N full-dimensional cones. One can then proceed by finding parameters $a_{C,i}$ which produce smooth fans, for all codimension 1 cones.

In dimension three we can use that the combinatorial types of simplicial fans are equivalent to the combinatorial types of triangulations of S^2 with at most N 2-simplices, which have been classified by R. Bowen and S. Fisk in *Generation of triangulations of the sphere* [2]. And Tadao Oda has done the classification of smooth two-dimensional fans which are minimal up to equivariant blowing-ups and the same in dimension three for at most eight rays in *Convex Bodies and Algebraic Geometry* [7]. We will explain and use those results later in chapter 3.

2.2.2. Generating Polytopes

The next step now is to find the polytopes. In the proof we used the edge lengths together with the normal fan to define the polytope. Here we will use the right-hand side of its inequality description $P(A, b) = \{x \mid Ax \leq b\}$ instead of the edge lengths, but after expressing the edge length in terms of the b-vector we can use the edge lengths to find all those b-vectors defining smooth lattice polytopes with the correct normal fan.

Right-Hand Side

For a given fan F we now need to find all possible b-vectors which yield smooth lattice polytopes with at most N lattice points and F as normal fan. Therefore define the set of all such vectors

$$\text{RHS}(F, N) = \{b \in \mathbb{Z}^m |\ \text{NF}_{P(A,b)} = F,\ |P(A, b) \cap \mathbb{Z}^d| \leq N\}$$

where the rows of A are the rays of F. In the following we want to define a polytope containing this set and enumerate all its lattice points.

Calculating the edge lengths

Lemma. Let $C = \text{cone}(n_1, n_2, \ldots, n_{d-1})$ be a codimension 1 cone and $r_1, r_2 \in \mathbb{Z}^d$ such that $C_1 = \text{cone}(C, r_1)$, $C_2 = \text{cone}(C, r_2) \in F$ are the full-dimensional cones containing C. Then the length $l(e)$ of the edge e in the polytope $P(A, b)$ dual to C can be expressed in terms of the right-hand side b as follows:

$$l(e) = b_{r_1} + b_{r_2} - \sum_{i=1}^{d-1} a_{C,i} b_{n_i}$$

where $a_{C,i}$ $(1 \leq i \leq d - 1)$ such that $r_1 + r_2 = \sum_{i=1}^{d-1} a_{C,i} n_i$.

Proof. The coefficients a_* are the same as in the proof of the theorem. Now let x_1 and x_2 be the vertices dual to C_1 and C_2 respectively. From the proof of the finiteness theorem we know $\langle u_d, -r_1 \rangle = 1$ and thus $l(e) = \langle x_2 - x_1, -r_1 \rangle$. Since $r_1 + r_2 = \sum_{i=1}^{d-1} a_{C,i} n_i$, this yields

$$
\begin{aligned}
l(e) &= \langle x_1, r_1 \rangle - \langle x_2, r_1 \rangle \\
&= b_{r_1} - \langle x_2, \sum_{i=1}^{d-1} a_{C,i} n_i - r_2 \rangle \\
&= b_{r_1} - \sum_{i=1}^{d-1} a_{C,i} \langle x_2, n_i \rangle + \langle x_2, r_2 \rangle \\
&= b_{r_1} - \sum_{i=1}^{d-1} a_{C,i} b_{n_i} + b_{r_2}
\end{aligned}
$$

$\square$

Approximating the Right-Hand Sides

Since we can express $l(e)$ linearly in terms of the b-vector we can now use bounds on the edge length to approximate $\text{RHS}(F, N)$. For a codimension 1 cone C, denote by e_C the edge in the polytope dual to C.

Lemma. For a given normal fan F in dimension d and a maximal number of lattice points N, every polytope defined by F and a vector from $\mathrm{RHS}(F, N)$ satisfies the following inequalities

$$l(e_C) \leq \frac{N - \sum_{i=1}^{d-1} a_{C,i}}{d} - 1 \qquad \forall C \in F^{(d-1)}$$

$$l(e_C) \geq 1 \qquad \forall C \in F^{(d-1)}$$

$$b_i = 0 \qquad \forall 1 \leq i \leq d$$

$$\sum_{C \in F^{(d-1)}} (l(e_C) - 1) \leq N - |F^{(d)}|$$

Using $l(e)$ as defined in the previous lemma, these inequalities define a polytope

$$\mathrm{B}(F, N) \supset \mathrm{RHS}(F, N)$$

Proof. We first show that $\mathrm{RHS}(F, N)$ satisfies the inequalities:

(i) We can now use an idea from the proof of the finiteness theorem. The convex hull of the thickened edge $c(e)$ contains $d(l(e)+1) + \sum_{i=1}^{d-1} a_{C,i}$ lattice points for an edge e with the parameters $a_{C,i}$, so we have the following bound for $l(e)$.

$$l(e) \leq \frac{N - \sum_{i=1}^{d-1} a_{C,i}}{d} - 1$$

(ii) The fan completely defines the combinatorial type of the polytope, so every codimension 1 cone must define a non-degenerate (i.e. $l(e) > 0$) edge in the polytope and since all vertices have integer coordinates the edge length must be at least one. This gives the second set of inequalities.

(iii) We now assume F to be lattice-transformed such that the first d rays are the unit vectors, this is possible because F is smooth. Since we do not care about translations, we can fix the first d right-hand sides to be zero.

(iv) The total number of lattice points on all edges of the polytope is the number of relative interior lattice points of each edge $(l(e) - 1)$ plus the number of vertices, so this sum must be less than the given bound N on the number of lattice points.

So all inequalities hold for any $b \in \mathrm{RHS}(F, N)$, they are all linear in terms of b and thus $\mathrm{B}(F, N)$ is a polyhedron.

Now to prove that this polyhedron is bounded we use a similar approach as we used to construct the fan from the edge-parameters. For every codimension 1 cone we have the following inequalities

$$1 \leq l(e) = b_{r_1} + b_{r_2} - \sum_{i=1}^{d-1} a_{C,i} b_{n_i} \leq N$$

12

Starting with the full-dimensional cone defined by the unit-vectors, where all b_i ($1 \leq i \leq d$) are zero, we can always find codimension 1 cones where only one of the b_* in the inequalities above is not yet bounded and thus bound it. We can iterate this until all b_* are bounded and thus $\mathrm{B}(F, N)$ is a polytope. $\square$

Enumeration

Now, given an inequality description of $\mathrm{B}(F, N)$ we can calculate all lattice vectors $b \in \mathrm{B}(F, N) \cap \mathbb{Z}^m$. Together with the matrix A, each one defines a smooth lattice polytope $P(A, b)$ with possibly too many lattice points. Thus, after removing all polytopes with more than N lattice points, we obtain a list of smooth lattice polytopes with normal fan F and at most N lattice points. The last step is to remove all polytopes that are lattice-isomorphic to another one. All these computations can be done with `polymake` [6] using interfaces to `4ti2` [1], `Latte macchiato` [4] and `normaliz2` [3].

3. Classification

After a short definition of blowing-ups, the sections 3.2 and 3.3 contain a detailed description of how to classify smooth two- and three-dimensional lattice polytopes with at most 12 lattice points based on the technique described above.

3.1. Blowing-ups

We want to use the results of Tadao Oda who classified smooth fans in dimension two and three which are minimal up to equivariant blowing-ups. This means that all normal fans we need can be generated by successive equivariant blowing-ups of the fans given in *Convex Bodies and Algebraic Geometry* by Tadao Oda [7]. But first we need to clarify what such a blowing-up is. In short, it means subdividing a cone and all cones containing it.

Definition. Let $\hat{C}$ be a cone in the fan F and $r \in \hat{C}$. The blowing-up $b(C, r)$ of a cone C at r is

$$b(C, r) = \{\text{cone}(\{r, C'\}) \mid C' \in F, C' \leq C, r \notin C'\} \qquad \text{if } r \in C$$
$$b(C, r) = \{C\} \qquad \text{if } r \notin C$$

where $C' \leq C$ means that C' is a face of C.
Now the **blowing-up** $b(F, r)$ of F at r is

$$b(F, r) = \bigcup_{C \in F} b(C, r)$$
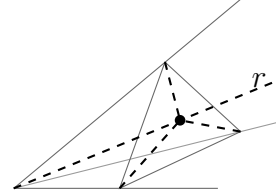
One can easily see that this is indeed again a fan.

A blowing-up is **equivariant** if $r = \sum_{s \in S} s$ for some cone $C = \text{cone}(S) \in F$, where S is minimal up to inclusion and all rays are primitive. We denote by $b(F, C)$ the blowing-up in cone C.
For a smooth fan, an equivariant blowing-up is again smooth. Since we want to classify smooth polytopes and thus smooth normal fans, we only have to consider equivariant blowing-ups.

Remark. Since we have a connection between the fan and the polytope one could ask what such a blowing-up of the normal fan corresponds to on the polytope side.
We first choose a cone in the fan, which is equivalent to choosing a face in the polytope, and then add the sum of all rays of the cone as new ray. This corresponds to adding a new facet by cutting-off that face of the polytope.

3.2. Smooth Polytopes in Dimension Two

3.2.1. Classifying the Normal Fans

As already mentioned we do not start with combinatorial types of fans, but with minimal smooth fans, and in dimension two there are only two such fans as given by the following theorem.

Minimal Fans

Theorem. (Tadao Oda [7]) *Every smooth fan in dimension two can, up to isomorphism, be created by finitely many successive equivariant blowing-ups of one of the following fans.*

(i) *The fan F_p representing the complex projective space $\mathbb{P}_2(\mathbb{C})$.*

$$F_p^{(1)} = \{a_0, a_1, a_2\} = \{(1,0), (0,1), (-1,-1)\}$$
$$F_p^{(d)} = \{\text{cone}(\{a_0, a_1\}), \text{cone}(\{a_1, a_2\}), \text{cone}(\{a_2, a_0\})\}$$

(ii) *The fans F_a representing the Hirzebruch surfaces for $a \in \mathbb{Z}$.*

$$F_a^{(1)} = \{a_0, a_1, a_2, a_3\} = \{(1,0), (0,1), (-1,-a), (0,-1)\}$$
$$F_a^{(d)} = \{\text{cone}(\{a_0, a_1\}), \text{cone}(\{a_1, a_2\}), \text{cone}(\{a_2, a_3\}), \text{cone}(\{a_3, a_0\})\}$$
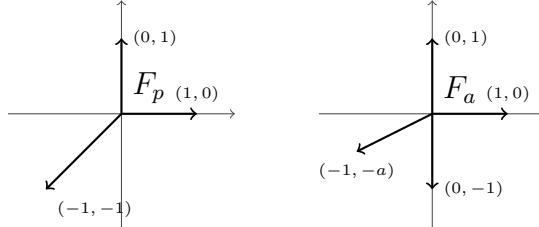
Tree of Blowing-ups

Instead of generating combinatorial types or finding parameters we have a different task now, namely equivariant blowing-ups. In dimension two this means choosing one full-dimensional cone $C = \text{cone}(r_1, r_2)$, inserting the ray $r = r_1 + r_2$ and replacing the cone C with $C_1 = \{r_1, r\}$ and $C_2 = \{r, r_2\}$.

So beginning with a fan F we can generate all normal fans with the following recursion:

1. For every full-dimensional cone $C \in F^{(d)}$:

a) Create the blowing-up $b(F, C)$,

b) If $b(F, C)$ has less than 12 full-dimensional cones: Start recursion.

Which is a depth-first search in a rather large rooted tree, having $1 + (3! + 4! + \cdots + 11!)/2! = 21977356$ nodes for up to 12 full-dimensional cones for the fan F_p, but lots or even most of those fans are isomorphic. So we need a way to reduce the number of nodes to consider:

We first assign an order to all full-dimensional cones. Now, since blowing up $b(F, C_i)$ in C_j and $b(F, C_j)$ in C_i will give the same fan, we can ignore all cones C_j with index $j < i$ in the subtree of the blowing-up $b(F, C_i)$. (When blowing up in cone C_i in a fan with k full-dimensional cones $C_1, \ldots, C_k$ we will index the two new cones with i and $k + 1$.) This reduces the amount of blowing-ups to 58785 for F_p and 35072 for F_a.

Parameters

The fan F_a and all further blowing-ups of it represent classes of infintely many fans with $a \in \mathbb{Z}$. But, from the proof of our main theorem, we know that for some upper bound N on the number of lattice points, there can be only finitely many of those. The proof even gives us a direct way to bound the parameter a.

Lemma. For the fan F_a and a maximal number of lattice points N for the polytope, we can restrict a to $0 \le a \le N$ and $a \ne 1$.

Proof. For a pair of two-dimensional cones $C_1 = \{r_1, r\}$ and $C_2 = \{r_2, r\}$ sharing a common ray r we can write $r_1 + r_2$ in terms of r, i.e. $r_1 + r_2 = a_r r$. Additionally we know $a_r \ge -l(e)$, where $l(e)$ is the length of the edge e dual to the ray r. Obviously $l(e) \le N$, so we have $a_r \ge -N$.

- For the cones $C_1 = \{(1, 0), (0, 1)\}$, $C_2 = \{(-1, -a), (0, 1)\}$ this yields

$$r_1 + r_2 = (1, 0) + (-1, -a) = (0, -a) = -a \cdot (0, 1)$$

and thus $-a = a_{(0,1),1} \ge -N$.

- And for $C_1 = \{(1, 0), (0, -1)\}$, $C_2 = \{(-1, -a), (0, -1)\}$ we have

$$r_1 + r_2 = (1, 0) + (-1, -a) = (0, -a) = a \cdot (0, -1)$$

and $a = a_{(0,-1),1} \ge -N$.

Because of the symmetry of the fan we can restrict to positive a and skip $a = 1$ since F_a then equals F_p after an equivariant blowing-up in the cone $C = \mathrm{cone}(\{a_0, a_2\})$. $\quad\square$

Parameters in the Blowing-up

We could find bounds on the parameter a of the Hirzebruch surface using an edge of the corresponding polytope, but when blowing up the fan, which means cutting off a vertex of the polytope, this edge may have different parameters and the bounds may be different too. We could of course do the same calculation for all blowing-ups, but a faster approach is to find a way to use the bounds we found for some fan F on all its blowing-ups.

Assume we have a pair of cones $C_1 = \{r_1, r\}$, $C_2 = \{r_2, r\} \in F$ around a ray r with $r_1 + r_2 = a_r r$, this means our bound is $a_r \geq -N$. Now, in the blowing-up $b(F, C_2)$, we have a new cone $C_2' = \{r_2', r\}$ with $r_2' = r + r_2$. This yields

$$r_1 + r_2' = a_r r - r_2 + r + r_2 = (a_r + 1)r$$

So we now have $a_r + 1 \geq -N$ and thus $a_r \geq -N - 1$. This means we have to loosen the bounds on a parameter by one if it appears as edge-parameter at one of the rays of the current cone.

Thus, after creating a new fan for every valid value of a for each class of fans, we now have all possible smooth normal fans for the smooth polytopes.

3.2.2. Creating the Polytopes

Now we can continue as in the general case, for every fan F we define the polytope $B(F, N)$ approximating $\mathrm{RHS}(F, N)$ and enumerate all its lattice points b. For some fan with rays A and a right-hand side b check if $|P(A, b) \cap \mathbb{Z}^2| \leq N$. And finally remove polytopes that are lattice-isomorphic to another one.

3.2.3. Results

We can now state the result of the computation:

Proposition. *There are 41 smooth lattice polygons with at most 12 lattice points.*

Vertices	3	4	5	6	7	8	≥ 9
Polygons	3	30	3	4	0	1	0

The complete list of polytopes can be found in the appendix.

3.3. Smooth Polytopes in Dimension Three

3.3.1. Classifying the Normal Fans

Minimal Fans

As in the two-dimensional case, we start with a list of smooth minimal fans from Tadao Oda [7].

Theorem. (Tadao Oda [7]) *Every smooth fan in dimension three, with at most eight rays, can, up to isomorphism, be created by finitely many successive equivariant blowing-ups of one of the* 19 *fans with labels* 3^4, $(3^2 4^3)'$, $(3^2 4^3)''$, 4^6, $3^2 4^3 6^2$, $3^1 4^3 5^3$, $4^5 5^2$, $4^6 6^2$, $3^2 4^4 7^2$, $3^3 4^1 5^1 6^3$, $3^2 4^2 5^2 6^2$, $3^1 4^4 5^1 6^2$, $3^2 4^1 5^4 6^1$, $(3^1 4^3 5^3 6^1)'$, $(3^1 4^3 5^3 6^1)''$, $(3^2 5^6)'$, $(3^2 5^6)''$, $(4^4 5^4)'$ *and* $(4^4 5^4)''$ *and weights as given in* Convex Bodies and Algebraic Geometry.

Remark.

- All drawings of such fans in the following chapter are stereographic projections of the corresponding triangulation of S^2.

- The labels give the occurence of vertices with different degrees in the triangulation, for example $3^2 4^3 6^2$ means 2 vertices of degree 3, 3 vertices of degree 4 and 2 vertices of degree 6. See the pictures later in this chapter.

Remark. With at most eight rays in the fan, from Eulers formula and the fact the the fan is simplicial we know that there are at most twelve full-dimensional cones in the fan and thus at most twelve vertices in the polytope. This is the main reason we chose 12 as upper bound for the number of lattice points in the polytope.

Even in the two-dimensional case, the blowing-up tree was huge so it comes in handy that we can use the two-dimensional classification now to reduce the number of fans we have to consider. From the 19 fans of the theorem there will be only five left after applying the now following criterion and it will help us a lot later during the computation of all blowing-ups.

Smooth Polygons

We use the smooth polygons we generated earlier to remove fans whose polytopes would have too many lattice points on the boundary. Let $\mathcal{P}_N(k)$ be the set of all smooth lattice k-gons with at most N lattice points. Define the following numbers:

$$l_N(k) = \min\{|P \cap \mathbb{Z}^2| \text{ for } P \in \mathcal{P}_N(k)\}$$
$$i_N(k) = \min\{|\operatorname{int}(P) \cap \mathbb{Z}^2| \text{ for } P \in \mathcal{P}_N(k)\}$$
$$b_N(k) = \min\{|P \cap \mathbb{Z}^2| - |\operatorname{int}(P) \cap \mathbb{Z}^2| \text{ for } P \in \mathcal{P}_N(k)\}$$

From the results of the previous section we can determine the values shown in the table below.

k	3	4	5	6	7	8	>9
$l_{12}(k)$	3	4	8	7	>12	12	>12
$i_{12}(k)$	0	0	1	1		4	
$b_{12}(k)$	3	4	7	6		8	

Denote by k_r the number of full-dimensional cones in F a ray r is contained in and let k_F be the maximal k_r over all rays r in F.

All vertices of the polytope defined by the full-dimensional cones around a ray r lie on the boundary of the facet defined by the ray r, so for every ray r there is a k_r-gon on the boundary of the polytope. Since the polytope is smooth, all these polygons have to be smooth, too. This yields the following lower bound for the number of lattice points, which depends solely on the combinatorial type of the fan.

Lemma. (**smooth lattice polygon criterion**) For a smooth lattice polytope P with the normal fan F.

$$|P \cap \mathbb{Z}^3| \geq |F^{(d)}| + (b_N(k_F) - k_F) + \sum_{r \in F^{(1)}} i_N(k_r)$$

Remark. This bound can be improved by considering the combinatorial type of the fan more precisely. However, for our computation it suffices.

Remaining Minimal Fans

Now we can check the fans from the theorem for the above criterion, the labels of the fans are exactly the numbers k_r^* we are looking for.

• First, consider all fans with eight rays and twelve full-dimensional cones. That are the fans labeled $4^6 6^2$, $3^2 4^4 7^2$, $3^3 4^1 5^1 6^3$, $3^2 4^2 5^2 6^2$, $3^1 4^4 5^1 6^2$, $3^2 4^1 5^4 6^1$, $(3^1 4^3 5^3 6^1)'$, $(3^1 4^3 5^3 6^1)''$, $(3^2 5^6)'$, $(3^2 5^6)''$, $(4^4 5^4)'$ and $(4^4 5^4)''$, which cannot be blown up as this would add two additional full-dimensional cones. From the labels one can see that they all have at least one ray contained in five, six or seven full-dimensional cones ($5^*, 6^*$ or 7^*), this means the corresponding polytope would have a five-, six- or seven-gon on the boundary. But since any smooth five- or six-gon has at least one interior lattice point and there is no smooth seven-gon with less than 13 lattice points at all, the polytope would have at least 13 lattice points on the boundary. So none of these fans admits a smooth lattice polytope with at most 12 lattice points.

• Now consider the fans labeled $4^5 5^2$ and $3^1 4^3 5^3$ with ten full-dimensional cones.

($4^5 5^2$) Here we have two non-incident rays each contained in five full-dimensional cones, which means we have two edge-disjoint five-gons in the polytope. But we know that there is no smooth five-gon with less than eight lattice points and thus any smooth polytope with this fan as normal fan would have at least 16 vertices.
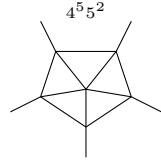
A smooth polytope corresponding to any blowing-up of this fan would have 12 vertices and either two six-gons or a five-gon and a six-gon and thus have at least 14 lattice points.

($3^1 4^3 5^3$) This fan contains three rays with five neighbors, so any polytope with this normal fan would have at least three lattice points in the relative interior of the facets and thus at least 13 lattice points in total. Any blowing-up would leave at least one of the five-gons unchanged and thus there would still be at least 13 lattice points, since we have now two additional vertices from the blowing-up and one in the interior lattice points in the five-gon.
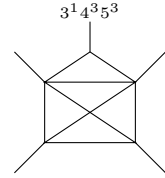
• By the above criterion, the fans numbered $3^4, (3^2 4^3)', (3^2 4^3)'', 4^6$ and $3^2 4^3 6^2$ could appear as normal fans for smooth lattice polytopes with at most 12 lattice points and will be the input to the algorithm.

This leaves us with the following five fans or classes thereof with rays and edge-parameters as drawn [7].

$(3^2 4^3)''$ $\quad \infty = (0,0,1)$

-1 $\qquad$ c

0 $\qquad\qquad$ b

$(-1,b,c)$ $\quad 0 \;—\; -1 \quad$ $(0,1,0)$

$0 \qquad b-c \quad -1$

$-1 \quad -c$

$(0,-1,-1)$ $\qquad\qquad$ $(1,0,0)$

$-b \qquad\qquad 0$

$-1 \;—\; 0$

$c-b \qquad\qquad\qquad -1$

4^6 $\qquad \infty = (0,0,1)$

b $\qquad\qquad$ c

0 $\qquad\qquad$ $-a$

$(-1,-a,c)$ $\quad 0 \;—\; 0 \quad$ $(0,1,0)$

$0 \qquad\qquad -a$

$-b \quad -c \qquad 0$

$(0,0,-1)$

$0 \qquad ab-c \quad -b \qquad 0$

$(0,-1,b)$ $\qquad a \qquad\qquad 0 \qquad$ $(1,0,0)$

$0 \;—\; 0$

$a \qquad\qquad\qquad 0$

$c-ab \qquad\qquad\qquad\qquad b$

$3^2 4^3 6^2$

$0 \qquad\qquad 0 \qquad\qquad\qquad \infty = (1,0,0) \qquad 0$

$a+1 \qquad\qquad 0 \qquad\qquad\qquad -a$

$(0,0,-1)$ $\quad 2a+1 \;—\; -2 \quad (0,-1,-a) \quad -2 \;—\; 2a-1 \quad (0,0,1)$

$-1 \quad a+1 \qquad 0 \qquad -a \quad -1$

$2 \qquad 0 \quad 0 \quad 0 \qquad 2$

$(0,1,-1)$ $\qquad 2 \;—\; -1 \quad (-1,2,-1) \quad -1 \;—\; 2 \qquad$ $(0,1,0)$

$2 \qquad\qquad 1 \qquad 1 \qquad\qquad 2$

$-1 \qquad\qquad 1 \qquad 1 \qquad\qquad -1$
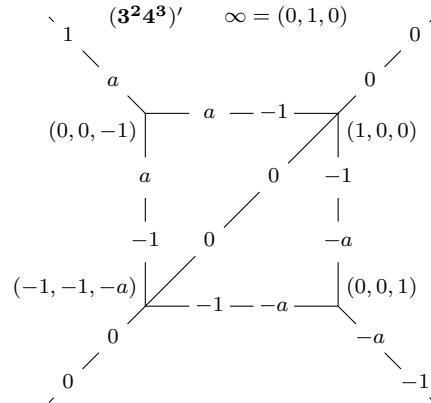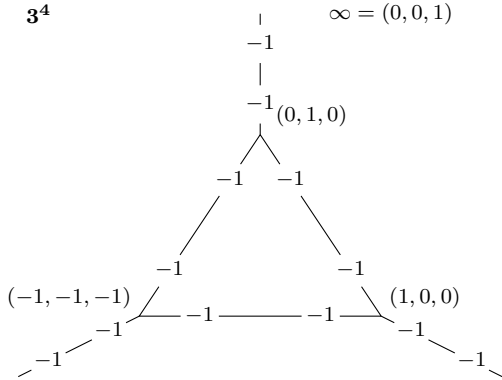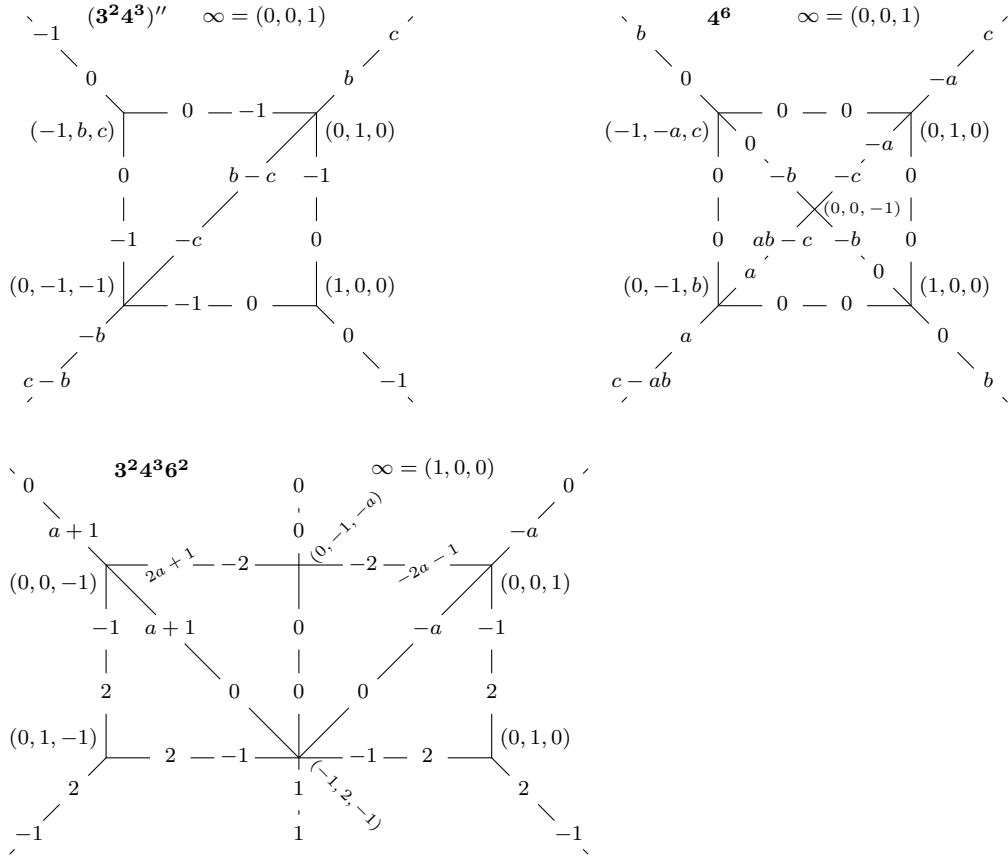
Tree of Blowing-ups

Again, the first step of the algorithm is to generate all possible normal fans by successive blowing-ups. There are now two possible cases, blowing up in a 3-dimensional cone or a 2-dimensional cone, which corresponds in the polytope to cutting off a vertex or an edge respectively, as shown in figure 2.

Figure 2: Blowing-up of a three-dimensional fan and the cutting-off in the corresponding polytope

Generating all blowing-ups successively gives again a huge rooted tree, but we now have more ways to reduce the number of nodes we have to check.

(1) We can again limit the depth of the tree since we want at most 12 full-dimensional cones and each blowing-up increases the number of such cones by two.

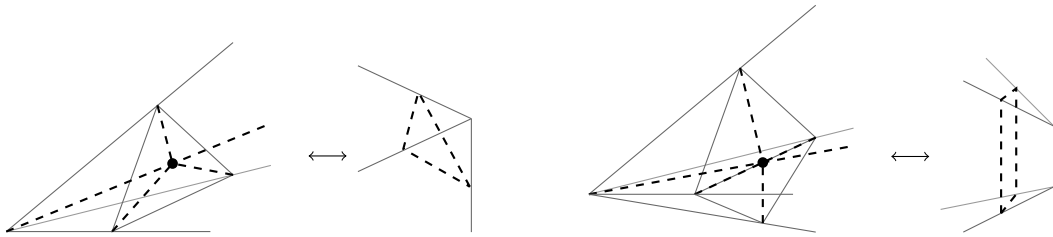(2) Blowing-ups in full-dimensional cones can again be done in any order. Let F contain k full-dimensional cones. When blowing up in the cone C_i the three new cones get the numbers $i, k+1, k+2$, so we can ignore all full-dimensional cones with index strictly less than i at all nodes in the subtree of this blowing-up.

(3) For codimension 1 cones there are two cases. Let C be a codimension 1 cone contained in two full-dimensional cones C_1 and C_2.

    (3a) Blowing up $b(F, C)$ in another codimension 1 cone C', which is contained in C_1 or C_2, will yield a different result than blowing up $b(F, C')$ in the cone C. All cones have to be considered in this case.

    (3b) Blowing up $b(F, C)$ in a codimension 1 cone C', which is not contained in C_1 or C_2, will yield the same result as blowing up $b(F, C')$ in C. So we can use an ordering on the codimension 1 cones to skip cones with lower index if they do not fall into case (3a).

(4) Blowing up in a full-dimensional cone and later in one of its codimension 1 faces yields the same rays and the same combinatorial type as blowing up in the codimension 1 cone first and then in on new codimension 1 cone as shown below.
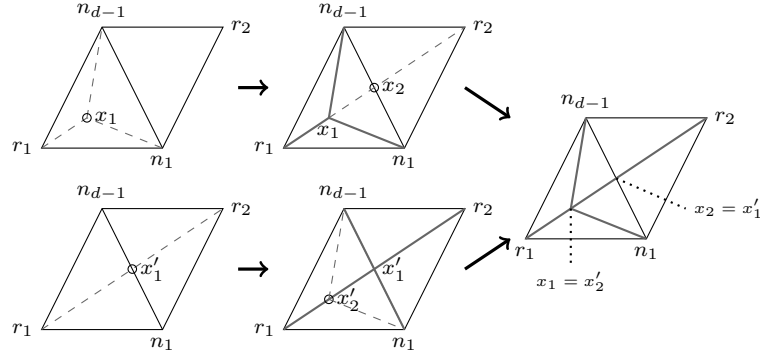
And we have the following new rays:

$$x_2 = \sum_{i=1}^{d-1} n_i \qquad\qquad x_1' = \sum_{i=1}^{d-1} n_i = x_2$$

$$x_1 = r_1 + \sum_{i=1}^{d-1} n_i \qquad\qquad x_2' = x_1' + r_1 = r_1 + \sum_{i=1}^{d-1} n_i = x_1$$

So only one of these cases has to be considered.

(5) As already mentioned, we can use the smooth lattice polygon criterion described earlier. Fans that violate this criterion will be considered for further blowing-ups,

but will not be kept for the later steps of the algorithm. We cannot cut off the tree here because the criterion is not monotone:

Assume we have a polytope with one vertex that is contained in three five-gons, then cutting-off this vertex will replace all with six-gons. So in total this new polytope will have two more vertices but may have less lattice points than the original polytope, since the smallest smooth five-gon has 8 lattice points in total and the smallest smooth six-gon is the hexagon with only 7 total lattice points.

This has been implemented with one variable for each full-dimensional and each codimension 1 cone that can have the values 'always ignore', 'ignore', 'consider', 'always consider'. We will create the blowing-up and continue the recursion only for cones with value 'consider' or 'always consider'. The specific cases are:

(2) Visited full-dimensional cones are marked 'ignore' for the subtree.

(3a) Visited codimension 1 cones are marked 'ignore' for the subtree if they haven't been marked 'always consider' or 'always ignore'.

(3b) Codimension 1 cones are marked 'always consider' if they were on 'ignore' before and are contained in one of the full-dimensional cones containing the codimension 1 cone of the current blowing-up.

(4) Codimension 1 cones are marked 'always ignore' if they were contained in the full-dimensional cone of the current blowing-up.

(x) Newly created cones are always set to 'consider'.

Parameters for the Minimal Fans

Most of the remaining minimal fans are actually classes of fans with parameters $a, b, c \in \mathbb{Z}$. Hence we have to find bounds for their parameters and find ways to use those bounds for their blowing-ups.

We can find bounds for their parameters analogously to the two-dimensional case: Whenever a parameter x appears as $a_{C,i} = x$ and $a_{C',j} = -x$ we know $-N \leq x \leq N$ and get:

$$
\begin{array}{rl}
[3^4] & \text{has no parameters,} \\
[(3^2 4^3)'] & 0 \leq a \leq N \text{ using the symmetry of the fan,} \\
[(3^2 4^3)''] & -N \leq b, c \leq N, \\
[4^6] & -N \leq a, b, c \leq N, \\
[3^2 4^3 6^2] & \text{here we use } a_{C,i} = 2a + 1 \text{ and } a_{C',j} = -2a - 1, \text{ so } \frac{-N-1}{2} \leq a \leq \frac{N-1}{2}.
\end{array}
$$

Parameters in the Blowing-up

In three dimensions we have to consider the following two different cases of blowing-ups:

- When blowing-up in a full-dimensional cone we can do the same as in the two-dimensional case.
  Denote by $C_1 = \{r_1, n_1, n_2\}$, $C_2 = \{r_2, n_1, n_2\}$ two full-dimensional cones containing a codimension 1 cone $C = \{n_1, n_2\}$ with $r_1 + r_2 = a_{C,1} n_1 + a_{C,2} n_2$. We blow up in the cone C_2, so the new full-dimensional cones containing C are C_1 and $C_2' = \{r_2', n_1, n_2\}$ with $r_2' = r_2 + n_1 + n_2$. We obtain

$$r_1 + r_2' = a_{C,1} n_1 + a_{C,2} n_2 + n_1 + n_2 = (a_{C,1} + 1) n_1 + (a_{C,2} + 1) n_2$$

  and this gives $a_{C,i} + 1 \geq -N$ and thus $a_{C,i} \geq -N - 1$. That means we have to loosen the bounds on a parameter by one if it appears at one of the codimension 1 cones contained in the cone of the current blowing-up, just as in the two-dimensional case.

- For the blowing-up in a codimension 1 cone choose C_1, C_2 and C as in the full-dimensional case. When blowing up in the cone C at $n = n_1 + n_2$ we get six new cones $C_1' = \{r_1, n_1, n\}$, $C_2' = \{r_2, n_1, n\}$ containing $C' = \{n_1, n\}$ and $C_1'' = \{r_1, n_2, n\}$, $C_2'' = \{r_2, n_2, n\}$ containing $C'' = \{n, n_2\}$. Now check the parameters at the new cone C':

$$r_1 + r_2 = a_{C,1} n_1 + a_{C,2} n_2 = a_{C,1} n_1 + a_{C,2}(n - n_1) = (a_{C,1} - a_{C,2}) n_1 + a_{C,2} n$$

  and at C'' we get equivalently:

$$r_1 + r_2 = a_{C,1} n_1 + a_{C,2} n_2 = a_{C,1}(n - n_2) + a_{C,2} n_2 = a_{C,1} n + (a_{C,2} - a_{C,1}) n_2$$

  So, in the blowing-up, we have found two codimension 1 cones each having one of the edge-parameters of the original fan, $a_{C',2} = a_{C,2}$ and $a_{C'',1} = a_{C,1}$. These edge-parameters of the two new cones C', C'' yield the same bounds for the parameters of the fan as the codimension 1 cone C in the original fan. Which means we can just pass on the bounds to the new blowing-up.

So we have again all neccessary bounds for the parameters of all fans and can create a list of completely defined smooth fans that serve as normal fans for our polytopes.

3.3.2. Creating the Polytopes

We can now continue as in the general and the two-dimensional case using `polymake` in combination with `4ti2`, `normaliz2` and `Latte macchiato` to compute all smooth polytopes with at most 12 lattice points.

3.3.3. Results

Running this algorithm for the remaining five minimal smooth fans yields the following result.

Proposition. *There are 33 smooth three-dimensional lattice polytopes with at most 12 lattice points.*

Vertices	4	6	8	≥ 10
Polytopes	2	25	6	0

Again the list of polytopes can be found in the appendix.

4. Final Comments

We now have a list of smooth lattice polytopes in dimension two and three with at most 12 lattice points. For dimension three the bound 12 may seem rather low and in fact none of the 33 lattice polytopes from this classification has an interior lattice point. So these polytopes may not bring great insight, but this is a first step and there are several points in the algorithm where it could be improved.

By implementing a different way to directly generate all smooth normal fans one could skip the big recursion calculating all blowing-ups, which is one of the most time-consuming steps in the classification algorithm. And one could then overcome the limits of at most 12 vertices and thus lattice points and dimension at most three.

The second point where one could work on is the calculation of lattice points of the polytope containing all right-hand sides. The dimension of this polytope is equal to the number of rays of the fan minus the dimension. So when we want to reach more complex smooth polytopes we will have rather high-dimensional polytopes containing the right-hand sides, so we might need better bounds for the right-hand sides and/or faster algorithms for computing the lattice points.

References

[1] 4ti2 team, *4ti2—a software package for algebraic, geometric and combinatorial problems on linear spaces*, Available at `http://www.4ti2.de`.

[2] Robert Bowen and Stephen Fisk, *Generation of triangulations of the sphere*, Mathematics of Computation **21** (1967), no. 98, 250–252.

[3] W. Bruns and B. Ichim, NORMALIZ. *computing normalizations of affine semigroups.*, With contributions by C. Söger. Available from `http://www.math.uos.de/normaliz`.

[4] Jesús A. De Loera, Raymond Hemmecke, Ruriko Yoshida, and Jeremy Tauzer, `lattE`, 2005, `http://www.math.ucdavis.edu/~latte/`.

[5] Alicia Dickenstein, Sandra Di Rocco, Christian Haase, Milena Hering, Diane Maclagan, Benjamin Nill, Paco Santos, Hal Schenck, and Greg Smith, *There are few smooth d-polytopes with n lattice points*, 2009.

[6] Ewgenij Gawrilow and Michael Joswig, *polymake: a framework for analyzing convex polytopes*, Polytopes — Combinatorics and Computation (Gil Kalai and Günter M. Ziegler, eds.), Birkhäuser, 2000, pp. 43–74.

[7] Tadao Oda, *Convex bodies and algebraic geometry, an introduction to the theory of toric varieties*, Ergebnisse der Mathematik und ihrer Grenzgebiete, Springer-Verlag, 1987.

[8] G.M. Ziegler, *Lectures on polytopes*, Graduate Texts in Mathematics, vol. 152, Springer-Verlag, 1995.

A. List of Smooth Polygons

(1,0)   (0,0)
(0,1)   (1,1)

(0,0)   (0,1)
(2,0)   (2,1)

(0,0)   (0,1)
(3,0)   (3,1)

(0,0)   (0,1)
(4,0)   (4,1)

(0,0)   (0,1)
(5,0)   (5,1)

(2,0)   (0,0)
(0,2)   (2,2)

(0,2)   (0,0)
(3,0)   (3,2)

(1,0)   (0,0)
(0,1)   (3,1)

(0,0)   (0,1)
(2,0)   (4,1)

(0,0)   (0,1)
(3,0)   (5,1)

(0,0)   (0,1)
(4,0)   (6,1)

(0,0)   (1,0)
(5,2)   (0,2)

(1,0)   (0,0)
(0,1)   (4,1)

(0,0)   (0,1)
(2,0)   (5,1)

(0,0)   (0,1)
(3,0)   (6,1)

(1,0)   (0,0)
(0,1)   (5,1)

(0,0)   (0,1)
(2,0)   (6,1)

(0,0)   (0,1)
(3,0)   (7,1)

(1,0)   (0,0)
(0,1)   (6,1)

(0,0)   (0,1)
(2,0)   (7,1)

(1,0)   (0,0)
(0,1)   (7,1)

(0,0)   (0,1)
(2,0)   (8,1)

(1,0)   (0,0)
(0,1)   (8,1)

(1,0)   (0,0)
(0,1)   (9,1)

(0,1)   (1,0)
(0,0)

(0,0)   (2,0)
(0,2)

(0,0)   (3,0)
(0,3)

(1,0)   (0,1)
(0,2)   (2,0)

(2,0)   (0,2)
(0,3)   (3,0)

(1,0)   (0,1)
(0,3)   (3,0)

(1,1)   (3,0)
(0,2)   (0,4)
(4,0)

(4,1)   (0,3)
(0,2)   (3,0)
(1,1)   (4,0)
(1,3)   (3,2)

(0,3)   (0,2)
(3,0)   (1,1)
(4,0)   (1,3)

(2,0)   (0,1)
(1,0)   (0,3)
(2,1)

(2,0)   (0,1)
(1,0)   (0,4)
(2,2)

(0,2)   (1,0)
(0,1)   (2,0)
(2,1)   (1,2)

(0,3)   (2,0)
(0,2)   (3,0)
(3,1)   (1,3)

(0,3)   (1,0)
(0,1)   (3,0)
(3,1)   (1,3)

(0,0)   (1,0)
(0,4)   (1,3)

(2,0)   (0,0)
(0,4)   (2,2)

(0,0)   (1,0)
(0,5)   (1,4)

B. List of Smooth Polytopes

(0,0,0) (0,1,0)
(0,0,1) (7,0,0)
(1,1,0) (1,0,1)

(0,0,0) (0,1,0)
(0,0,1) (6,0,0)
(1,1,0) (1,0,1)

(0,0,0) (0,1,0)
(0,0,1) (6,0,0)
(2,1,0) (1,0,1)

(0,0,0) (0,1,0)
(0,0,1) (5,0,0)
(1,1,0) (1,0,1)

(0,0,0) (0,1,0)
(0,0,1) (5,0,0)
(2,1,0) (1,0,1)

(0,0,0) (0,1,0)
(0,0,1) (5,0,0)
(3,1,0) (1,0,1)

(0,0,0) (0,1,0)
(0,0,1) (4,0,0)
(1,1,0) (1,0,1)

(0,0,0) (0,1,0)
(0,0,1) (5,0,0)
(2,1,0) (2,0,1)

(0,0,0) (0,1,0)
(0,0,1) (4,0,0)
(2,1,0) (1,0,1)

(0,0,0) (0,1,0)
(0,0,1) (4,0,0)
(3,1,0) (1,0,1)

(0,0,0) (0,1,0)
(0,0,1) (4,0,0)
(4,1,0) (1,0,1)

(0,0,0) (0,1,0)
(0,0,1) (3,0,0)
(1,1,0) (1,0,1)

(0,0,0) (0,1,0)
(0,0,1) (4,0,0)
(2,1,0) (2,0,1)

(0,0,0) (0,1,0)
(0,0,1) (3,0,0)
(2,1,0) (1,0,1)

(0,0,0) (0,1,0)
(0,0,1) (4,0,0)
(3,1,0) (2,0,1)

(0,0,0) (0,1,0)
(0,0,1) (3,0,0)
(3,1,0) (1,0,1)

(0,0,0) (0,1,0)
(0,0,1) (2,0,0)
(2,1,0) (1,0,1)

(0,0,0) (0,1,0)
(0,0,1) (3,0,0)
(3,1,0) (2,0,1)

(0,0,1) (0,1,0)
(1,0,0) (0,0,0)

(0,0,0) (2,0,0)
(0,2,0) (0,0,2)

(1,0,0) (0,1,0)
(1,0,1) (0,1,1)
(0,2,0) (2,0,0)

(2,0,0) (0,2,0)
(2,0,1) (0,2,1)
(0,3,0) (3,0,0)

(0,1,0) (1,0,0)
(0,0,0) (0,0,1)
(1,0,1) (0,1,1)

(0,0,0) (0,0,1)
(2,0,0) (0,2,0)
(2,0,1) (0,2,1)

(0,0,0) (1,0,0)
(0,1,0) (0,0,2)
(1,0,2) (0,1,2)

(0,0,0) (1,0,0)
(0,1,0) (0,0,3)
(1,0,3) (0,1,3)

(0,0,0) (0,0,1)
(2,0,0) (0,2,0)
(1,0,1) (0,1,1)

(0,1,0) (0,0,0)
(0,0,1) (0,1,1)
(3,0,0) (3,1,0)
(1,0,1) (1,1,1)

(0,1,0) (0,0,0)
(0,0,1) (0,1,1)
(2,0,0) (3,1,0)
(1,0,1) (2,1,1)

(0,1,0) (0,0,0)
(0,0,1) (0,1,1)
(2,0,0) (2,1,0)
(1,0,1) (1,1,1)

(0,1,0) (0,0,0)
(0,0,1) (0,2,1)
(2,0,0) (2,1,0)
(1,0,1) (1,2,1)

(1,1,0) (0,1,0)
(0,0,0) (1,0,0)
(0,0,1) (1,0,1)
(0,1,1) (1,1,1)

(0,1,0) (0,0,0)
(0,0,1) (0,1,1)
(2,0,0) (2,1,0)
(2,0,1) (2,1,1)

28

C. `polymake` **extension**

C.1. Parameterized Fans (`pfan.rules`)

```
# a parameterized class of fans
declare object ParameterFan<Rational> {

# maximal number of lattice points
property POINT_LIMIT : Int;

# dimension of the fan
property DIM : Int;

# incidence sets (of the rays) defining the full-dimensional cones or
# vertices of the polytope
property VERTICES : Array<Set>;

# number of vertices
property N_VERTICES : Int;

# incidence sets (of the vertices) defining the codimension 1 cones or
# edges between vertices of the polytopes
property EDGES : Array<Set>;

# number of edges
property N_EDGES : Int;

# rays of the fan
property RAYS : Matrix<Rational>;

# number of rays
property N_RAYS : Int;

# number of different parameters
property N_PARAMETER : Int;

#  rows with positions in RAYS matrix, parameter number and coefficient
property PARAMETER_POS : Matrix<Int>;

# lower and upper bound for each parameter (rowwise)
property PARAMETER_BOUNDS : Matrix<Int>;

# searches full-dimensional cones sharing a common codimension 1 cone
rule EDGES : VERTICES , N_VERTICES , DIM {
  my $vert = $this->VERTICES;
  my @edges = ();
  local $_;
  # check all possible pairs
  foreach (all_subsets_of_k(2,0..($this->N_VERTICES-1))) {
    my $r = $vert->[$_->[0]] * $vert->[$_->[1]];
    if ($#$r == ($this->DIM - 2)) {
      push @edges, new Set($_);
    }
  }
  $this->EDGES = new Array<Set>(@edges);
}

rule DIM : RAYS {
  $this->DIM = $this->RAYS->cols;
}

rule N_EDGES : EDGES {
  $this->N_EDGES = $#{$this->EDGES} + 1;
```

```
}

rule N_VERTICES : VERTICES {
  $this->N_VERTICES = $#{$this->VERTICES} + 1;
}

rule N_RAYS : RAYS {
  $this->N_RAYS = $this->RAYS->rows;
}


# main method   creating the list of polytopes for this fan
# calls all necessary other functions
# returns:        the polytopes as perl array of Polytope objects
# input:          integer value determining the print- or debug-level
user_method POLYTOPES {
  my ($this,$print) = @_;
  my @poly = ();
  print "Creating blowing-ups ...\n" if($print);
  my @blowups = $this->create_blowups($print);
  print "Done creating blowing-ups.\n" if($print);
  my $i=1;
  print "Generating polytopes ...\n" if($print);
  foreach my $pfan (@blowups) {
    print "blowup ",$i," of ",scalar(@blowups),"\n" if ($print);
    my @fans = $pfan->getfans();
    my $j = 1;
    foreach my $fan (@fans) {
      print "\tfan ",$i,":",$j," of ",scalar(@blowups),":",
                scalar(@fans),"\n" if ($print);
      my $num = $fan->N_POLYTOPES;
      if ($num > 0) {
        push @poly, $fan->POLYTOPE_LIST;
      }
      $j++;
    }
    $i++;
  }
  print "Done generating polytopes.\n" if($print);
  print "Found ",scalar(@poly), " polytopes including duplicates.\n" if($print);
  print "Removing duplicates ...\n" if($print);
  my @final_poly = ();
  outer: for (my $i = scalar(@poly)-1; $i >= 0; $i--) {
    my $p = @poly[$i];
    for (my $j = $i-1; $j >= 0; $j--) {
      my $q = @poly[$j];
      # reduce neccessary isomorphy checks by checking number of vertices first
      if ($p->N_VERTICES == $q->N_VERTICES) {
        if (lattice_isomorphic_smooth_polytopes($p,$q)) {
          next outer;
        }
      }
    }
    push @final_poly, $p;
  }

  print "Found ",scalar(@final_poly), " non-isomorphic polytopes with this normal fan.\n";
  return @final_poly;
}

# starts the recursion generating all blowing-ups returning them in an perl array
# printlimit specifies up to which depth to print messages
user_method create_blowups {
  my ($fan,$printlimit) = @_;
  # incidence vectors storing which vertices and edges should be blown-up in the tree,
  # zero vectors at the beginning
  my $vertexlist = new Vector<Int>($fan->POINT_LIMIT);
```

```perl
    my $edgelist = undef;
    $edgelist = new Vector<Int>(($fan->DIM * $fan->POINT_LIMIT)/2) if ($fan->DIM > 2);

    my @blowups = recursion($fan,$vertexlist,$edgelist,0,$printlimit);
    return @blowups;
}

# creates a new fan for every combination of parameter values and
# returns the list of fans as perl array
user_method getfans {
  my $bf = shift;
  my $rays = $bf->RAYS;
  my @fans = ();
  if ($bf->N_PARAMETER > 0) {
    my $npar = $bf->N_PARAMETER;
    my $pb = $bf->PARAMETER_BOUNDS;
    # vector storing the current parameter values
    my $par = new Vector<Int>($pb->col(0));
    my $cont = 1;
    do {
      my $r = new Matrix<Rational>($rays);
      # add the parameter values to every position it occurs
      foreach my $pos (@{$bf->PARAMETER_POS}) {
        $r->($pos->[0],$pos->[1]) += $par->[$pos->[2]] * $pos->[3];
      }
      push @fans,new Fan<Rational>(POINT_LIMIT=>($bf->POINT_LIMIT),DIM=>($bf->DIM),
          RAYS=>$r,VERTICES=>($bf->VERTICES),EDGES=>($bf->EDGES));

      # generate next vector of parameter values
      # increase first parameter and check bounds,...
      $par->[0] ++;
      for (my $i = 0; $i < $npar; $i++) {
        if($par->[$i] > $pb->[$i]->[1]) {
          if($i == $npar-1) {
            $cont = 0;
          } else {
            $par->[$i] = $pb->[$i]->[0];
            $par->[$i+1]++;
          }
        }
      }
    } while($cont == 1);
  } else {
    push @fans,new Fan<Rational>(POINT_LIMIT=>($bf->POINT_LIMIT),DIM=>($bf->DIM),
        RAYS=>$rays,VERTICES=>($bf->VERTICES),EDGES=>($bf->EDGES));
  }
  return @fans;
}

# main recursion creating the blowing-ups
sub recursion {
  my ($fan,$vertexlist,$edgelist,$depth,$printlimit) = @_;
  my @b = ();

  if ($fan->DIM > 2) {
    # in dimension > 2 check smooth polygon criterion
    if (checkpolygons($fan) <= $fan->POINT_LIMIT) {
      push @b, $fan;
    }
  } else {
    push @b, $fan;
  }

  # if blowing-up does not exceed maximal number of vertices
  if ($fan->N_VERTICES <= $fan->POINT_LIMIT - ($fan->DIM-1)) {
    # blowing-up of full-dimensional cones
```

31

```perl
      for(my $i = 0; $i < $fan->N_VERTICES; $i++) {
        if ($vertexlist->[$i] < 1) {
          print "do vertexblowup:",$depth,"-",$i,"\n"if ($depth < $printlimit);
          my ($newfan,$elist) = do_vertex_blowup($fan,$i,$edgelist);
          push @b, recursion($newfan,(new Vector<Int>($vertexlist)),
                             $elist,$depth+1,$printlimit);
          $vertexlist->[$i]=1;
        }
      }
      # if dim > 2 blowing-up of codim 1 cones
      if($fan->DIM > 2) {
        for (my $i = 0; $i < $fan->N_EDGES; $i++) {
          if ($edgelist->[$i] < 1) {
            print "do edgeblowup:",$depth,"-",$i,"\n" if ($depth < $printlimit);
            my ($newfan,$vlist,$elist) = do_edge_blowup($fan,$i,$vertexlist,$edgelist);
            push @b, recursion($newfan,$vlist,$elist,$depth+1,$printlimit);
            if($edgelist->[$i] == 0) {
              $edgelist->[$i] = 1;
            }
          }
        }
      }
    }
  }
  return @b;
}


# create the new fan after the blowing-up of full-dim cone $i
sub do_vertex_blowup {
  my ($fan,$i,$edgelist) = @_;
  # set of rays defining the vertex
  my $list = $fan->VERTICES->[$i];
  my $nrays = $fan->N_RAYS;
  my $elist = undef;
  $elist = new Vector<Int>($edgelist) if (defined($edgelist));

  # create and add the new ray
  my $ray = new Vector<Rational>($fan->DIM);
  local $_;
  foreach (@$list) {
    $ray += $fan->RAYS->[$_];
  }
  my $newrays = $fan->RAYS / $ray;

  # prepare edges array, vertices array and array with positions for new vertices
  my @newedges = @{$fan->EDGES};
  my @pos = ($i);
  my @vertices = @{$fan->VERTICES};
  for (my $k = 0; $k < $fan->DIM - 1; $k++) {
    push @pos, ($fan->N_VERTICES+$k);
    push @vertices, new Set();
  }
  local $_;
  foreach (all_subsets_of_k(2,@pos)) {
    push @newedges, (new Set($_));
  }

  my $k = 0;
  my $newvertices = new Array<Set>(@vertices);
  my $newedges = new Array<Set>(@newedges);
  # find neighbors of vertex we are cutting off, correct the edges
  # and create new vertices
  for (my $j = 0; $j < $fan->N_EDGES; $j++) {
    my $edge = $fan->EDGES->[$j];
    if ($edge->contains($i))  {
      my $v = ($edge - $i)->[0];
      # new vertex
```

32

```perl
      $newvertices->[@pos[$k]] = ($fan->VERTICES->[$v] * $list) + $nrays;
      # set new edge endpoints
      $newedges->[$j] = new Set($v,@pos[$k]);
      # set neighboring edges to always ignore,
      # we do vertex -> neighboring edge via edge -> edge
      $elist->[$j] = 2 if (defined($edgelist));
      $k++;
    }
  }

  my $newfan;
  # add parameters for the new ray and loosen bounds if neccessary
  my $par = new Matrix<Int>();
  if ($fan->N_PARAMETER > 0) {
    my @param = @{$fan->PARAMETER_POS};
    local $_;
    my $p_indices = new Vector<Int>($fan->N_PARAMETER);
    # find parameters appearing in the rays defining the currently blown-up cone
    foreach (@{$fan->PARAMETER_POS}) {
      if ($list->contains($_->[0])) {
        push @param, (new Vector<Int>($nrays,$_->[1],$_->[2],$_->[3]));
        $p_indices->[$_->[2]]=1;
      }
    }
    $par = new Matrix<Int>(@param);
    # loosen bounds on all parameters found earlier
    my $pbounds = new Matrix<Int>($fan->PARAMETER_BOUNDS);
    for (my $j = 0; $j < $fan->N_PARAMETER; $j++) {
      if ($p_indices->[$j] > 0) {
        $pbounds->[$j]->[0] --;
        $pbounds->[$j]->[1] ++;
      }
    }
    # create new fan with all modified data
    $newfan = new ParameterFan<Rational>(POINT_LIMIT=>$fan->POINT_LIMIT,
        DIM=>$fan->DIM,VERTICES=>$newvertices,EDGES=>$newedges,RAYS=>$newrays,
        PARAMETER_POS=>$par,PARAMETER_BOUNDS=>$pbounds,N_PARAMETER=>$fan->N_PARAMETER);
  } else {
    $newfan = new ParameterFan<Rational>(POINT_LIMIT=>$fan->POINT_LIMIT,
        DIM=>$fan->DIM,VERTICES=>$newvertices,EDGES=>$newedges,RAYS=>$newrays,
        PARAMETER_POS=>$par,N_PARAMETER=>$fan->N_PARAMETER);
  }
  return ($newfan,$elist);
}

# create the new fan after the blowing-up of full-dim cone $i
sub do_edge_blowup {
  my ($fan,$i,$vertexlist,$edgelist) = @_;
  my $vlist = new Vector<Int>($vertexlist);
  my $elist = new Vector<Int>($edgelist);
  # indices of the two vertices of the edge
  my $edge = $fan->EDGES->[$i];
  my $e1 = $edge->[0];
  my $e2 = $edge->[1];
  # sets of the rays defining these vertices
  my $v1 = $fan->VERTICES->[$e1];
  my $v2 = $fan->VERTICES->[$e2];
  # two rays defining this edge (as set and separate)
  my $er = $v1*$v2;
  my $er1 = $er->[0];
  my $er2 = $er->[1];
  # total number of vertices and rays
  my $nv = $fan->N_VERTICES;
  my $nrays = $fan->N_RAYS;

  # add new ray
```

```perl
  my $newrays = $fan->RAYS / ($fan->RAYS->[$er->[0]] + $fan->RAYS->[$er->[1]]);

  # new edgearray
  my @newedges = @{$fan->EDGES};
  # create edges to and between the two vertices that will be put at the end
  push @newedges , new Set($nv,$nv+1);
  push @newedges , new Set($e1,$nv);
  push @newedges , new Set($e2,$nv+1);
  my $k = 0;

  # create new vertices
  my $newvertices = new Array<Set>(@{$fan->VERTICES},new Set(), new Set());
  $newvertices->[$e1] = $v1 - $er2 + $nrays;
  $newvertices->[$e2] = $v2 - $er2 + $nrays;
  $newvertices->[$nv] = $v1 - $er1 + $nrays;
  $newvertices->[$nv+1] = $v2 - $er1 + $nrays;

  # set the new vertices  to 'consider' for later blowing-ups
  $vlist->[$e1] = 0;
  $vlist->[$e2] = 0;

  # create the new edges and update all old edges
  # we skip all just added edges and the cut-off edge
  my $newedges = new Array<Set>(@newedges);
  for (my $j = 0; $j < $fan->N_EDGES; $j++) {
    next if ($i==$j);
    my $e = $fan->EDGES->[$j];
    if ($e->contains($e1) || $e->contains($e2)) {
      my $v = ($e - $edge)->[0];
      if (!($fan->VERTICES->[$v]->contains($er1))) {
        $newedges->[$j] = new Set($v,$nv) if ($e->contains($e1));
        $newedges->[$j] = new Set($v,$nv+1) if ($e->contains($e2));
      }
      $elist->[$j] = -1 if ($elist->[$j]==1);
    }
  }

  my $newfan;
  # check if the new ray has parameters
  my $par = new Matrix<Int>();
  if ($fan->N_PARAMETER > 0) {
    my @param = @{$fan->PARAMETER_POS};
    local $_;
    # search for parameters for the rays defining the new one
    foreach (@{$fan->PARAMETER_POS}) {
      if ($er->contains($_->[0])) {
        push @param, (new Vector<Int>($nrays,$_->[1],$_->[2],$_->[3]));
      }
    }
    $par = new Matrix<Int>(@param);
    $newfan = new ParameterFan<Rational>(POINT_LIMIT=>$fan->POINT_LIMIT,
            DIM=>$fan->DIM,VERTICES=>$newvertices,EDGES=>$newedges, RAYS=>$newrays,
            PARAMETER_POS=>$par,PARAMETER_BOUNDS=>$fan->PARAMETER_BOUNDS,
            N_PARAMETER=>$fan->N_PARAMETER);
  } else {
    $newfan = new ParameterFan<Rational>(POINT_LIMIT=>$fan->POINT_LIMIT,
            DIM=>$fan->DIM,VERTICES=>$newvertices,EDGES=>$newedges, RAYS=>$newrays,
            N_PARAMETER=>$fan->N_PARAMETER);
  }
  return ($newfan,$vlist,$elist);
}

# check the smooth polygon lower bound for the number of lattice points
sub checkpolygons{
  my ($fan) = @_;
  my $rays = $fan->N_RAYS;
```

```perl
  # results from the 2-dimensional classification ,
  # "13" means not possible for <= 12 lattice points
  # FIXME for more than POINT_LIMIT > 12
  my $innercounts  = new Vector <Rational >(’0 0 1 1 13 4 13 13’);
  my $bordercounts = new Vector <Rational >(’3 4 7 6 13 8 13 13’);

  # collect degrees of all rays (number of vertices they appear in)
  my $sizes = new Vector <Int >( ’0 ’ x $rays );
  local $_;
  foreach (@{$fan ->VERTICES}) {
    foreach my $x (@$_) {
      $sizes ->[$x] ++;
    }
  }

  # find maximal degree , and sum up all interior lattice points
  my $count = 0;
  my $max = 0;
  local $_;
  foreach (@$sizes) {
    if ($_ > $max) {
      $max = $_;
    }
    next if ($_ > 8);
    $count += $innercounts ->[$_ -3];
  }
  if ($max > 8) {
    return 13;
  }
  # add boundary lattice points for the largest polygon
  $count += $bordercounts ->[$max -3];
  $count += $fan ->N_VERTICES - $max;
  return $count;
}

}
```

C.2. Fans (`fan.rules`)

```perl
# a completely defined fan
declare object Fan<Rational > {

# maximal number of lattice points
property POINT_LIMIT : Rational;

# dimension of the fan
property DIM : Int;

# incidence sets (of the rays) defining the full - dimensional cones or
# vertices of the polytope
property VERTICES : Array <Set >;

# number of vertices
property N_VERTICES : Int;

# incidence sets (of the vertices) defining the codimension 1 cones or
# edges between vertices of the polytopes
property EDGES : Array <Set >;

# number of edges
property N_EDGES : Int;

# rays of the fan
```

```
property RAYS : Matrix<Rational>;

# number of rays
property N_RAYS : Int;

# polytope approximating the set of valid right-hand sides
property RHS_POLYTOPE : Polytope<Rational>;

# set of right-hand side vectors defining polytopes
# with at most POINT_LIMIT lattice points
property RHS_VECTORS : Set<Vector<Integer>>;

# set of right-hand side vectors defining polytopes
# with at most POINT_LIMIT lattice points
property N_POLYTOPES : Integer;

# returns a perl array of all polytopes with this normal fan and
# at most POINT_LIMIT lattice points
user_method POLYTOPE_LIST {
  my $fan = shift;
  my @list = ();
  # create polytope for every rhs vector
  foreach my $rhs (@{$fan->RHS_VECTORS}) {
    my $ineq = (new Vector<Rational>($rhs)) | ($fan->RAYS);
    my $p = new Polytope<Rational>(INEQUALITIES=>$ineq);
    #my $v = $p->N_VERTICES;
    #my $lp = $p->N_LATTICE_POINTS;
    push @list, $p;
  }
  return @list;
}


# searches full-dimensional cones sharing a common codimension 1 cone
rule EDGES : VERTICES , N_VERTICES , DIM {
  my $vert = $this->VERTICES;
  my @edges = ();
  local $_;
  # check all possible pairs
  foreach (all_subsets_of_k(2,0..($this->N_VERTICES-1))) {
    my $r = $vert->[$_->[0]] * $vert->[$_->[1]];
    if ($#$r == ($this->DIM - 2)) {
      push @edges, new Set($_);
    }
  }
  $this->EDGES = new Array<Set>(@edges);
}

rule DIM : RAYS {
  $this->DIM = $this->RAYS->cols;
}

rule N_EDGES : EDGES {
  $this->N_EDGES = $#{$this->EDGES} + 1;
}

rule N_VERTICES : VERTICES {
  $this->N_VERTICES = $#{$this->VERTICES} + 1;
}

rule N_RAYS : RAYS {
  $this->N_RAYS = $this->RAYS->rows;
}

rule N_POLYTOPES : RHS_VECTORS {
  $this->N_POLYTOPES = $#{$this->RHS_VECTORS}+1;
}
```

```
# searches in RHS_POLYTOPE for RHS-vectors defining polytopes
# with at most POINT_LIMIT lattice points
rule RHS_VECTORS : RHS_POLYTOPE , RAYS , POINT_LIMIT , DIM {
  my $valid = $this->RHS_POLYTOPE;
  my $limit = $this->POINT_LIMIT;
  my $dim = $this->DIM;
  my $rays = $this->RAYS;

  # check the polytope for feasibility and number of lattice points first
  if ($valid->FEASIBLE == 0) {
    $this->RHS_VECTORS=new Set<Vector<Integer>>();
    return;
  }
  if ($valid->N_LATTICE_POINTS == 0) {
    $this->RHS_VECTORS=new Set<Vector<Integer>>();
    return;
  }

  # remove first coordinate (homogenized)
  my $rhs = $valid->LATTICE_POINTS->minor(All,range(1,$rays->rows()-$dim));
  my @rhs = @$rhs;
  @rhs = sort @rhs;

  my @good_rhs = ();
  my $i=0;
  my $sched2;

  while (@rhs > 0) {
    # create new RHS vector and inequality matrix
    my $x = ((zero_vector<Int>($dim)) | shift(@rhs));
    my $ineq = (new Vector<Rational>($x)) | $rays;

    # define polytope and check number of lattice points
    my $p = new Polytope<Rational>(INEQUALITIES=>$ineq);

    if ($p->N_LATTICE_POINTS <= $limit) {
      # found a new polytope
      push @good_rhs , $x;
      $i++;
    } else {
      # we only keep RHS-vectors with at least one entry with smaller
      # absolute value than the current vector
      my @newrhs = ();
      point: for my $v (@rhs) {
        for(my $j = 0; $j < $rays->rows()-$dim; $j++) {
          if (abs($v->[$j]) < abs($x->[$j+$dim])) {
            push @newrhs,$v;
            next point;
          }
        }
      }
      @rhs = @newrhs;
    }
  }
  $this->RHS_VECTORS=new Set<Vector<Integer>>(@good_rhs);
}

# creates the polytope approximating the set of right-hand sides
rule RHS_POLYTOPE : RAYS , VERTICES , EDGES , POINT_LIMIT , N_EDGES , N_VERTICES {
  my $vert = $this->VERTICES;
  my $edges = $this->EDGES;
  my $rays = $this->RAYS;
  my $dim = $rays->cols();
  my $r = $rays->rows();
  my $limit = $this->POINT_LIMIT;
```

```perl
    # stores the sum of all edge-lengths
    my $totallength = new Vector<Rational>($r+1);

    # first stores l(e) >= 1 inequalities, second one inequalities for l(e)<=...
    my @ineq = ();
    my @l_ineq = ();

    # calculate all edge lengths and create inequalities
    foreach (@$edges) {
      # vertices defining the edge
      my $fromvert = $vert->[$_->[0]];
      my $tovert = $vert->[$_->[1]];

      # rays defining the edge (n_1,...,n_{d-1})
      my $neighbors = $fromvert * $tovert;

      # rays limiting this edge (r_1,r_2)
      my $from = $fromvert - $neighbors;
      my $to = $tovert - $neighbors;
      $from = $from->[0];
      $to = $to->[0];

      # calculate inequalities
      my ($edgelimit,$pointlimit) = edgelimits($rays,$from,$to,$limit,$neighbors);

      $totallength += $pointlimit;
      push @ineq, $edgelimit;
      push @l_ineq, $pointlimit;
    }

    # right-hand side for total length inequality
    $totallength->[0] = $limit + $this->N_EDGES - $this->N_VERTICES;
    push @l_ineq, $totallength;

    my $ineq = new Matrix<Rational>(\@ineq);
    my $l_ineq = new Matrix<Rational>(\@l_ineq);

    # for all rays with all entries positive we know that the right-hand side must
    # be postive and vice versa (we want the negative RHS-vectors)
    my $signs = new Matrix<Rational>($r,$r+1);
    my @signs = ();
    row: for (my $i = 0; $i < $r; $i++) {
      my $sign = 0;
      for (my $j = 0; $j < $dim; $j++) {
        if ($rays->($i,$j) < 0) {
          next row if ($sign > 0);
          $sign = -1;
        } elsif ($rays->($i,$j) > 0) {
          next row if ($sign < 0);
          $sign = 1;
        }
      }
      $signs->($i,$i+1) = -$sign;
    }

    # put all inequalities together and remove the first d columns after the rhs entry
    # because those variables are always zero (we do not want translations)
    $ineq = $ineq / $signs;
    $ineq = $ineq->minor(All,range(0,0)+range($dim+1,$r));
    $l_ineq = $l_ineq->minor(All,range(0,0)+range($dim+1,$r));
    my $limited = (new Matrix<Rational>($ineq)) / (new Matrix<Rational>($l_ineq));

    $this->RHS_POLYTOPE = new Polytope<Rational>( INEQUALITIES=>$limited );
}
```

```
# calculates the inequalities in terms of the rhs vector given by 1<=l(e)<=...
sub edgelimits {
  my ($rays, $from, $to, $limit, $nb) = @_;
  my $dim = $rays->cols();
  # limiting rays
  my $r1 = $rays->row($to);
  my $r2 = $rays->row($from);
  # rays defining the edge
  my $N = $rays->minor($nb,All);

  # determine edge-parameters a_{C,i}
  $N = new Matrix<Rational>(transpose($N));
  my $v = new Vector<Rational>($r1+$r2);
  my $a = lin_solve($N,$v);
  my $sum = 0;
  # and sum them up for the right-hand side
  $sum += $a->[$_] for (0 .. ($dim-2));

  # vectors storing the edgelength >= 1 and <=... inequalities
  my $edgelimit = new Vector<Rational>($rays->rows() + 1);
  my $pointlimit = new Vector<Rational>($rays->rows() + 1);

  # rhs of the inequalities
  $edgelimit->[0] = (-1);
  $pointlimit->[0] = (($limit - $sum)/$dim-1);

  # set coefficients
  for (my $i = 0; $i < $dim-1; $i++) {
    my $k = $nb->[$i] + 1;
    $edgelimit->[$k] = (-1) * $a->[$i];
    $pointlimit->[$k] = $a->[$i];
  }
  $edgelimit->[$to+1] = 1;
  $pointlimit->[$to+1] = -1;
  $edgelimit->[$from+1] = 1;
  $pointlimit->[$from+1] = -1;

  return ($edgelimit,$pointlimit);
}

}
```

C.3. Additional Script (`checkiso`)

```
# removes all lattice isomorphic polytopes from a list
# only the first one of an isomorphy class is kept
sub checkiso {
  my (@list) = @_;
  my @res = ();
  outer: for (my $i = scalar(@list)-1; $i >= 0; $i--) {
    my $p = @list[$i];
    for (my $j = $i-1; $j >= 0; $j--) {
      my $q = @list[$j];
      if ($p->N_VERTICES == $q->N_VERTICES) {
        if ($p->N_LATTICE_POINTS == $q->N_LATTICE_POINTS) {
          next outer if (lattice_isomorphic_smooth_polytopes($p,$q));
        }
      }
    }
    push @res, $p;
  }
  return @res;
}
```

C.4. Minimal Fans as `polymake` files

C.4.1. Dimension Two

Projective Space

```
<?xml version="1.0" encoding="utf-8"?>

<object name="P" type="polytope::ParameterFan&lt;Rational&gt;" version="2.9.7"
tm="4b0d8c7b" xmlns="http://www.math.tu-berlin.de/polymake/#3">
  <property name="POINT_LIMIT" value="12" />
  <property name="DIM" value="2" />
  <property name="VERTICES">
    <m>
      <v>0 1</v>
      <v>1 2</v>
      <v>0 2</v>
    </m>
  </property>
  <property name="RAYS">
    <m>
      <v>1 0</v>
      <v>0 1</v>
      <v>-1 -1</v>
    </m>
  </property>
  <property name="N_RAYS" value="3" />
  <property name="N_VERTICES" value="3" />
  <property name="N_PARAMETER" value="0" />
</object>
```

Hirzebruch surfaces

```
<?xml version="1.0" encoding="utf-8"?>

<object name="H" type="polytope::ParameterFan&lt;Rational&gt;" version="2.9.7"
tm="4af96f7b" xmlns="http://www.math.tu-berlin.de/polymake/#3">
  <property name="POINT_LIMIT" value="12" />
  <property name="DIM" value="2" />
  <property name="VERTICES">
    <m>
      <v>0 1</v>
      <v>0 2</v>
      <v>1 3</v>
      <v>2 3</v>
    </m>
  </property>
  <property name="RAYS">
    <m>
      <v>1 0</v>
      <v>0 1</v>
      <v>0 -1</v>
      <v>-1 0</v>
    </m>
  </property>
  <property name="N_RAYS" value="4" />
  <property name="N_VERTICES" value="4" />
  <property name="PARAMETER_POS">
    <m>
      <v>3 1 0 1</v>
    </m>
```

```
      </property >
  <property name="PARAMETER_BOUNDS">
    <m>
      <v>0 12</v>
    </m>
  </property >
  <property name="N_PARAMETER" value="1" />
</object >
```

C.4.2. Dimension Three

Fan 3^4

```
<?xml version="1.0" encoding="utf-8"?>

<object name="f" type="polytope::ParameterFan&lt;Rational&gt;" version="2.9.7"
tm="4b0d8c1d" xmlns="http://www.math.tu-berlin.de/polymake/#3">
  <property name="POINT_LIMIT" value="12" />
  <property name="DIM" value="3" />
  <property name="VERTICES">
    <m>
      <v>0 1 2</v>
      <v>0 1 3</v>
      <v>0 2 3</v>
      <v>1 2 3</v>
    </m>
  </property >
  <property name="RAYS">
    <m>
      <v>1 0 0</v>
      <v>0 1 0</v>
      <v>0 0 1</v>
      <v>-1 -1 -1</v>
    </m>
  </property >
  <property name="N_RAYS" value="4" />
  <property name="N_VERTICES" value="4" />
  <property name="N_PARAMETER" value="0" />
</object >
```

Fan $(3^2 4^2)'$

```
<?xml version="1.0" encoding="utf-8"?>

<object name="f" type="polytope::ParameterFan&lt;Rational&gt;" version="2.9.7"
tm="4b0d571c" xmlns="http://www.math.tu-berlin.de/polymake/#3">
  <property name="POINT_LIMIT" value="12" />
  <property name="DIM" value="3" />
  <property name="VERTICES">
    <m>
      <v>0 1 2</v>
      <v>0 1 3</v>
      <v>0 2 4</v>
      <v>0 3 4</v>
      <v>1 2 4</v>
      <v>1 3 4</v>
    </m>
  </property >
  <property name="RAYS">
    <m>
      <v>1 0 0</v>
      <v>0 1 0</v>
```

```
      <v>0  0  1</v>
      <v>0  0  -1</v>
      <v>-1  -1  0</v>
    </m>
  </property>
  <property name="N_RAYS" value="5" />
  <property name="N_VERTICES" value="6" />
  <property name="PARAMETER_POS">
    <m>
      <v>4  2  0  -1</v>
    </m>
  </property>
  <property name="PARAMETER_BOUNDS">
    <m>
      <v>0  12</v>
    </m>
  </property>
  <property name="N_PARAMETER" value="1" />
</object>
```

Fan $(3^2 4^2)''$

```
<?xml version="1.0" encoding="utf-8"?>

<object name="f" type="polytope::ParameterFan&lt;Rational&gt;" version="2.9.7"
tm="4b0d8bd2" xmlns="http://www.math.tu-berlin.de/polymake/#3">
  <property name="POINT_LIMIT" value="12" />
  <property name="DIM" value="3" />
  <property name="VERTICES">
    <m>
      <v>0  1  2</v>
      <v>0  1  3</v>
      <v>0  2  3</v>
      <v>1  2  4</v>
      <v>1  3  4</v>
      <v>2  3  4</v>
    </m>
  </property>
  <property name="RAYS">
    <m>
      <v>1  0  0</v>
      <v>0  1  0</v>
      <v>0  0  1</v>
      <v>0  -1  -1</v>
      <v>-1  0  0</v>
    </m>
  </property>
  <property name="N_RAYS" value="5" />
  <property name="N_VERTICES" value="6" />
  <property name="PARAMETER_POS">
    <m>
      <v>4  1  0  1</v>
      <v>4  2  1  1</v>
    </m>
  </property>
  <property name="PARAMETER_BOUNDS">
    <m>
      <v>-12  12</v>
      <v>-12  12</v>
    </m>
  </property>
  <property name="N_PARAMETER" value="2" />
</object>
```

Fan 4^6

```
<?xml version="1.0" encoding="utf-8"?>

<object name="f" type="polytope::ParameterFan&lt;Rational&gt;" version="2.9.7"
tm="4b0d8a3c" xmlns="http://www.math.tu-berlin.de/polymake/#3">
  <property name="POINT_LIMIT" value="12" />
  <property name="DIM" value="3" />
  <property name="VERTICES">
    <m>
      <v>0 1 2</v>
      <v>0 1 3</v>
      <v>0 2 4</v>
      <v>0 3 4</v>
      <v>1 2 5</v>
      <v>1 3 5</v>
      <v>2 4 5</v>
      <v>3 4 5</v>
    </m>
  </property>
  <property name="RAYS">
    <m>
      <v>1 0 0</v>
      <v>0 1 0</v>
      <v>0 0 1</v>
      <v>0 0 -1</v>
      <v>0 -1 0</v>
      <v>-1 0 0</v>
    </m>
  </property>
  <property name="N_RAYS" value="6" />
  <property name="N_VERTICES" value="8" />
  <property name="PARAMETER_POS">
    <m>
      <v>5 1 0 -1</v>
      <v>4 2 1 1</v>
      <v>5 2 2 1</v>
    </m>
  </property>
  <property name="PARAMETER_BOUNDS">
    <m>
      <v>-12 12</v>
      <v>-12 12</v>
      <v>-12 12</v>
    </m>
  </property>
  <property name="N_PARAMETER" value="3" />
</object>
```

Fan $3^2 4^3 6^2$

```
<?xml version="1.0" encoding="utf-8"?>

<object name="f" type="polytope::ParameterFan&lt;Rational&gt;" version="2.9.7"
tm="4b0d42cd" xmlns="http://www.math.tu-berlin.de/polymake/#3">
  <property name="POINT_LIMIT" value="12" />
  <property name="DIM" value="3" />
  <property name="VERTICES">
    <m>
      <v>0 1 2</v>
      <v>0 1 6</v>
      <v>0 2 5</v>
      <v>0 3 4</v>
      <v>0 3 5</v>
```

```
        <v>0  4  6</v>
        <v>1  2  6</v>
        <v>2  5  6</v>
        <v>3  4  6</v>
        <v>3  5  6</v>
      </m>
  </property>
  <property name="RAYS">
      <m>
        <v>1  0  0</v>
        <v>0  1  0</v>
        <v>0  0  1</v>
        <v>0  0  -1</v>
        <v>0  1  -1</v>
        <v>0  -1  0</v>
        <v>-1  2  -1</v>
      </m>
  </property>
  <property name="N_RAYS" value="7" />
  <property name="N_VERTICES" value="10" />
  <property name="PARAMETER_POS">
      <m>
        <v>5  2  0  -1</v>
      </m>
  </property>
  <property name="PARAMETER_BOUNDS">
      <m>
        <v>-6  5</v>
      </m>
  </property>
  <property name="N_PARAMETER" value="1" />
</object>
```